ECE452C\_Fall-2018

Project-02\_Team-7

Team member:

Yunjuan Wang

Yinbin Ma

Qiteng Wu

Video Link:

Task allocation:

1. Image capture: Yinbin Ma
2. Image analysis and processing: Qiteng Wu
3. Robot moving: Yunjuan Wang
4. Debug the code and do the experiment: Yunjuan Wang, Yinbin Ma, Qiteng Wu

Difficulties:

1. It takes significant trial and testing to find out the proper thresholds for robot moving in one iteration.
2. We tried to let one wheel move forward and let the other wheel move backward but failed. Thus, we make the assumption that the coordinates remain the same when the robot rotate.

Steps of tuning the sensors:

Since only our right infrared sensor works well and there is something wrong with our left infrared sensor, we only detect if there is obstacle in front of our right infrared sensor.

Code & Comments:

1. Configure Environment

utils.py

Function: Robot configuration for the convenience of writing code calls later.

1. import RPi.GPIO as GPIO
2. import picamera as pc
3. from AlphaBot import AlphaBot
4. import numpy as np
6. Ab = AlphaBot()
7. Ab.stop()
9. S1 = 27
10. S2 = 22
11. DL = 19
13. GPIO.setmode(GPIO.BCM)
14. GPIO.setwarnings(False)
15. GPIO.setup(S1, GPIO.OUT)
16. GPIO.setup(S2, GPIO.OUT)
17. GPIO.setup(DL, GPIO.IN, GPIO.PUD\_UP)
19. PS1 = GPIO.PWM(S1, 50)
20. PS1.start(50)
21. PS2 = GPIO.PWM(S2, 50)
22. PS2.start(50)
24. def set\_servo1(angle): # vertical
25. PS1.ChangeDutyCycle(12.5 - 10.0 \* **float**(angle) / 180)
27. def set\_servo2(angle): # horizontal
28. PS2.ChangeDutyCycle(12.5 - 10.0 \* **float**(angle) / 180)
29. Image capture:

capture.py

Function: Capture a specific number *n* of images. Return these images.

1. from utils import pc, set\_servo1, set\_servo2, np
2. import time
3. weight = np.array([0.2126, 0.7152, 0.0722])
5. def init():
6. set\_servo2(0)
7. time.sleep(0.5)
9. def capture(n):
10. r = pc.PiResolution(480, 320).pad()
11. imgs = np.zeros([n, 480\*320], np.uint8)
12. c = pc.PiCamera(resolution=r)
13. **try**:
14. **for** i in range(n):
15. set\_servo2(180/(n-1)\*i)
16. time.sleep(0.5)
17. tmp = np.zeros(480\*320\*3, np.uint8)
18. c.capture(tmp, 'rgb')
19. tmp = np.uint8(np.dot(tmp.reshape([-1, 3]), weight))
20. imgs[i, :] = tmp
21. finally:
22. c.close()
23. **return** list(imgs)
24. Image analysis and processing:

light.py

Function: Calculate the sum of pixel value in every image. Return the angle that the sum of pixel value in that image is maximum.

1. from utils import np
3. def Light(imgs):
4. maxNum = 0
5. indexMax = 0
6. **for** index, img in enumerate(imgs):
7. count = np.sum(img)
8. **if** count > maxNum:
9. maxNum = count
10. indexMax = index
12. **return** indexMax\*(180/(len(imgs)-1))
13. Robot moving:

lightmove.py

Function: If the robot turns left, right wheel moves and left wheel stops. If the robot turns right, left wheel moves and right wheel stops. Record the distance of the corresponding wheel when robot rotates based on the sensor of our speeding test module. Given a specific angle, we can calculate the distance that the corresponding wheel has to move. Then the robot goes straightforward in a specific distance and stop. The robot will repeat from capturing images to adjust moving direction until it detects obstacle (we assume the light source is an obstacle) and stop. We use infrared sensor to detect obstacles. Since only our right infrared sensor works well and there is something wrong with our left infrared sensor, we only detect if there is obstacle in front of our right infrared sensor. We print out the coordinates when the robot stops.

1. from \_\_future\_\_ import print\_function
2. import math
3. import time
4. import sys
5. from utils import Ab, GPIO
7. cntl = 7
8. cntr = 8
9. EncL = 0
10. EncR = 0
11. DL = 19
13. def updateEncoderL(channel):
14. global EncL
15. EncL += 1
17. def eprint(\*args, \*\*kwargs):
18. print(\*args, file=sys.stderr, \*\*kwargs)
20. def updateEncoderR(channel):
21. global EncR
22. EncR += 1
24. GPIO.setup(cntr, GPIO.IN)
25. GPIO.setup(cntl, GPIO.IN)
26. GPIO.add\_event\_detect(cntr, GPIO.BOTH, updateEncoderR)
27. GPIO.add\_event\_detect(cntl, GPIO.BOTH, updateEncoderL)
29. def detect\_obst():  # detect light source as obstacles.
30. DL\_status = GPIO.input(DL)
31. **return** DL\_status == 1
32. def followlightmove(angle):
33. global EncL, EncR
34. ini\_encl = EncL
35. ini\_encr = EncR
36. speed = 30
38. print("Starting.")
40. time.sleep(0.5)
41. d = 5  # the distance between two wheel
42. threshold = 3 \* math.pi
43. k = 0.2
45. **if** angle > 90:
46. Ab.setMotor(0, speed)
47. count = 0
48. **while** True:
49. distance\_l = **float**(EncL - ini\_encl) / 40 \* 2 \* math.pi \* 1.5
50. **if** count % 200 == 0:
51. eprint("l:", distance\_l)
52. **if** distance\_l >= d \* abs(angle - 90) \* 2 \* math.pi / 360:
53. **break**
54. count += 1
55. Ab.stop()
57. **else**:
58. Ab.setMotor(-speed, 0)
59. count = 0
60. **while** True:
61. distance\_r = **float**(EncR - ini\_encr) / 40 \* 2 \* math.pi \* 1.5
62. **if** count % 200 == 0:
63. eprint("r:", distance\_r)
64. **if** distance\_r >= d \* abs(angle - 90) \* 2 \* math.pi / 360:
65. **break**
66. count += 1
67. Ab.stop()
69. af\_encl = EncL
70. af\_encr = EncR
72. Ab.setMotor(-speed, speed+2)
74. count = 0
75. straight\_distance = 0  # distance that the robot move straightforward.
77. **while** detect\_obst():
78. straight\_distance = **float**(EncL - af\_encl + EncR - af\_encr) / 2 / 40 \* 2 \* math.pi \* 1
79. **if** count % 100 == 0:
80. eprint("straight:", straight\_distance)
81. count += 1
82. **if** straight\_distance > threshold:
83. **break**
85. Ab.stop()
87. y = math.cos(angle \* 2 \* math.pi / 360) \* straight\_distance
88. x = -math.sin(angle \* 2 \* math.pi / 360) \* straight\_distance / 1.5
89. **return** x, y

Test:

app.py

1. import capture
2. import light
3. import lightmove
4. import time
5. import math
6. import numpy as np
8. x, y = 0, 0  # initialize the original coordinates.
9. n = 7
10. # try:
11. **while** lightmove.detect\_obst():
12. capture.init()
13. img = capture.capture(7)
14. angle = light.Light(img)
15. print angle
16. time.sleep(1)
17. dx, dy = lightmove.followlightmove(angle)
19. x += dx  # update coordinates.
20. y += dy
22. print "angle: {} degree\nx: {}\ny: {}".format(angle, dx, dy)
23. #
24. # except KeyboardInterrupt:
25. #     lightmove.Ab.stop()
27. angle = math.atan2(y, x)
28. # calculate SE(2) based on angle and moving distance in x-axis and y-axis
29. se2 = [math.cos(angle), -math.sin(angle), x,
30. math.sin(angle), math.cos(angle), y,
31. 0, 0, 1]
33. se2 = np.array(se2).reshape([3, 3])
35. print "SE(2) for flashing light:\n", se2